



Hojas >	Ej.1	Ej.2	Ej.3	Ej.4	
6	1	1	2	2	
Calif. >	B	B <sup>-</sup>	B	B <sup>+</sup>	Final: A <sup>+</sup>

Todas las respuestas se consideran válidas solo si están debidamente justificadas.  
Entregar cada ejercicio en hojas separadas.

### Ejercicio 1

Un host con dirección IP 192.168.0.1 ha establecido una conexión TCP con el host 152.92.27.21. En algún hop intermedio, un sniffer defectuoso pudo capturar parcialmente los paquetes TCP intercambiados durante la comunicación:

Source	Destination	Info
192.168.0.1	152.92.27.21	7170 > 2073 [SYN] Seq=0 Ack=0 Len=0 <?>
192.168.0.1	152.92.27.21	7170 > 2073 [ACK] Seq=1 Ack=51 Len=0 <?>
152.92.27.21	192.168.0.1	2073 > 7170 [ACK] Seq=51 Ack=101 Len=50 <?>
192.168.0.1	152.92.27.21	7170 > 2073 [ACK] Seq=102 Ack=202 Len=0

- Sabiendo que 152.92.27.21 ha enviado 200 bytes de datos hasta el momento, completar con cero o más paquetes las secciones faltantes (marcadas con <?>) de forma tal que ambos extremos no repitan estados, a excepción de ESTABLISHED.
- Extender la captura proponiendo una serie de paquetes que hagan que el socket del host 152.92.27.21 atraviese los siguientes estados:

ESTABLISHED → FIN WAIT 1 → FIN WAIT 2 → TIME WAIT → CLOSED

### Ejercicio 2

En la siguiente tabla, se pueden ver algunas variables que tiene una conexión TCP recién establecida. En dicha conexión el receiver anuncia una *Advertised Window* cada vez más chica hasta que en el 5to RTT, el emisor se ve obligado a frenar el envío de datos. Luego de un instante, llegan ACKs anunciando una ventana más grande, lo que hace que, en el 8vo RTT, la RWND aumente dejando de liderar la conexión.

RTT	SSTHRESH	RWND	Last Bytes Sent	Flight Size
1	64KB	32KB	4KB	4KB
2	64KB	28KB	12KB	8KB
3	64KB	20KB	28KB	16KB
4	64KB	4KB	32KB	4KB
5	64KB	0KB	32KB	0KB
6	64KB	0KB	32KB	0KB
7	64KB	0KB	32KB	0KB
8	64KB	64KB	...	...
9	...	...	...	...

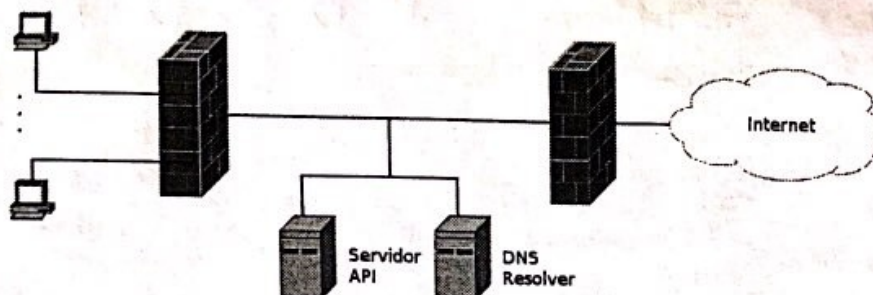
- Enumerando por RTT, continúe el valor de las variables del control de congestión asumiendo que no se producen pérdidas de paquetes, hasta que se transmitan un total de 50KB de datos (incluidos los datos ya enviados). Incluya los valores de CWND a lo largo de toda la traza hasta la llegada de los últimos ACKs.



- b. 150ms después de terminado el envío anterior llegan de la capa superior 20KB para enviar y el  $RTT = 100ms$ . Suponiendo que hay pérdida por congestión de todos los paquetes al alcanzar ráfagas de 10KB o más, y que el receptor siempre anuncia una *Advertised Window* de 64KB. ¿Cuántos bytes se deben retransmitir debido a pérdidas por congestión? Justifique.

### Ejercicio 3

En la figura se muestra la organización de red de una compañía. En la DMZ se encuentra un servidor brindando una API en el puerto 29995. Dicho servidor es el único al que se puede tener acceso desde internet. Los empleados desde la red interna pueden visitar sitios web usando HTTP o HTTPS, pero sólo pueden realizar consultas DNS que sean recursivas, al Resolver que se encuentra en la DMZ.

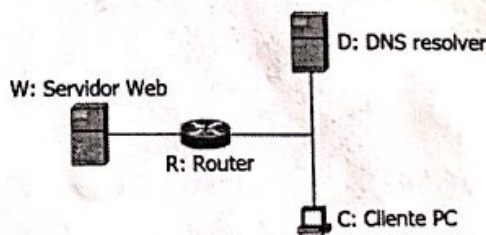


- Definir las reglas de ambos firewalls que cumplan con los requerimientos mencionados.
- Para poder realizar consultas DNS de manera segura se desea implementar un sistema que permita garantizar la autenticidad del Resolver DNS. Explicar qué mensajes deben firmarse y cómo se firman, aclarando dónde deben instalarse los certificados digitales.
- ¿Qué otras propiedades cumple el sistema criptográfico diseñado en el ítem (b.)? (i.e.: confidencialidad / integridad / no repudio / autenticación / disponibilidad) Justificar por sí o por no para cada una.

### Ejercicio 4

En la topología que se muestra en la figura, un cliente C se enciende en la red y quiere descargar la página web <http://www.ar/index.html> del servidor W. Esto desencadena varios paquetes en los dispositivos de la red. Suponiendo que el host C ya tiene una IP configurada manualmente:

- Enumere todos los mensajes de capa de aplicación.
- Enumere todos los segmentos de capa de transporte.
- Enumere todos los paquetes de capa de red.



Asumir lo siguiente:

- Todos los enlaces son segmentos Ethernet.
- Todas las máquinas tienen sus caches ARP vacías.
- D tiene en su caché DNS los datos correspondientes a [www.ar](http://www.ar).
- El cliente recién enciende, de tal manera que sus caches están vacías (DNS, ARP).
- Los contenidos de <http://www.ar/index.html> más los headers HTTP entran en un único segmento TCP.

*Hint: Se dice que una trama es de una capa cuando contiene información de protocolos de HASTA esa capa*



19.06.18

- ① Para simplificar la notación, solo A para simbolizar 192.168.0.1 <sup>ok</sup> y B para 152.92.27.21. <sup>ok</sup>  
 Además, no solo los números de puertos 7170 > 2073 <sup>ok</sup>  
 en la info para que sea más simple, pero deberían  
 estar (7170 > 2073 para A → B, y al revés para  
 B → A) <sup>ok</sup>

a)

	Src.	Dest.	Info.
1	<u>A</u>	B	[SYN] Seq=0, Ack=0, Len=0 ✓
2	B	<u>A</u>	[SYN+ACK] Seq=0, Ack=1, Len=0 ✓
3	<u>A</u>	B	[ACK] Seq=1, Ack=1, Len=0 ✓
4	B	<u>A</u>	[ACK] Seq=1, Ack=1, Len=50 ✓
5	<u>A</u>	B	[ACK] Seq=1, Ack=51, Len=0
6	<u>A</u>	B	[ACK] Seq=1, Ack=51, Len=100.
	<del>A</del>	<del>B</del>	<del>[ACK] Seq=1, Ack=101, Len=50</del>
7	B	<u>A</u>	[ACK] Seq=51, Ack=101, Len=50 ✓
8	<u>A</u>	B	[ACK] Seq=101, Ack=101, Len=1
9	B	<u>A</u>	[ACK] Seq=101, Ack=102, Len=100.
10	B	<u>A</u>	[FIN] Seq=201, Ack=102, Len=0
11	<u>A</u>	B	[ACK] Seq=102, Ack=202, Len=0

Después de enviar 1, A está en SYN-SENT  
 y B en LISTEN. Después de 2, A en  
 SYN-SENT y B en SYN-RCVD. Después de  
 3, A en ESTABLISHED y B en SYN-RCVD.  
 B estará en ESTABLISHED cuando  
 le llegue el paquete. <sup>ok</sup>

• A = 192.168.0.1 <sup>ok</sup>  
 • B = 152.92.27.21 <sup>ok</sup>  
 • • = puertos capturados <sup>ok</sup>

→ sigue



En el resto de los paquetes ambas partes en ESTABLISHED, hasta el seq. 10; luego de enviarlo B permanece en FIN-WAIT-1, y luego de enviar el 11 A queda en CLOSE-WAIT, y B en FIN-WAIT-2 cuando lo recibe.

En síntesis, ningún extremo repite su estado por más de un RTT (sin contar ESTABLISHED), ya que lo cambiam al recibir una respuesta. Además, no se vuelve a estados anteriores:

A: SYN-SENT, ESTABLISHED, CLOSE-WAIT

B: LISTEN, SYN-RCVD, ESTABLISHED, FIN-WAIT1

b) Ya quedó en FIN-WAIT2 cuando recibió el último paquete de A.

Seq.	Src.	Dest.	Yulo
12	A	B	[FIN] Seq=102, Ack=202, Len=0
13	B	A	[ACK] Seq=202, Ack=103, Len=0

Antes de sto, B ya está en FIN-WAIT2. Al recibir [FIN] de A pasa a TIME-WAIT y envía un [ACK].

Luego de aproximadamente 120 segundos, pasa a CLOSED: ok

ESTABLISHED → FIN-WAIT1 → FIN-WAIT2 → TIME-WAIT → CLOSED

- Nota: Para el ejercicio dice un cliente de notación y me referí a A como el ~~7170~~ puerto 7170 de 192.168.0.1 y ok  
B como el 2073 de 152.92.27.21 y ok ~~Además se usa~~  
~~192.168.0.1~~



② Suponga:  $1W = 2 \cdot SMSS = 4KB$ . Las unidades de la tabla son KB. Suponga  $LW = 1 \cdot SMSS = 2KB$ .

a)

RTT	CWND	SSTHRESH	RWND	LBS	Flight Size
1	4	64	32	4	4
2	8	}	28	12	8
3	16		20	28	16
4			4	32	4
5			0	32	0
6			0	32	0
7			0	32	0
8	4	64	64	36	4
9	8	64	64	44	8
10	16	64	64	50	6

Como el emisor no envió datos en los RTT 5 a 7, se lo considera "IDLE" y se reinicia la ventana, pero por más de un  $RTO = 2 \cdot RTT$ . Por esto  $CWND = 1W = 4KB$  en el 8vo RTT.

b) (Suponga que el problema de la congestión con ráfagas de 10KB ~~que se produce al inicio de la transmisión~~ ~~se resuelve a partir de ahora~~)

RTT	CWND	SSTHRESH	RWND	LBS (desde uno)	Flight Size
1	16	64	64	16	20
2	16	64	64	16	16
3	2	8 x 10	64	2	2
4	4	8	64	6	4
5	8	8	64	14	8
6	10	8	64	24	10
7	10	8	64	24	10
8	2	5	64	16	2
9	4	5	64	20	4

timeout →

timeout →

Sigue →



Retx	RTT	CWND	SSTHRESH	RWND	LWS	Flight Size
1	8	2	5	64	16	2
2	9	4	5	64	20	

Como esto comienza  $150 \text{ ms} = 1,5 \text{ RTT}$  después de la conexión, no se considera que el emisor está "idle" en ese tiempo, y comienza el CWND:  $16 \text{ KB}$  ( $1,5 \cdot \text{RTT} < \text{RTO} = 2 \cdot \text{RTT}$ )

En los RTT 1 y 6, hay en vuelo  $10 \text{ KB}$  o más, por lo que se pierden. En sus respectivos RTT siguientes, 2 y 7, no hay omisión de datos porque se considera que la vía de comunicación está llena ( $\text{MAX WINDOW} = \text{Flight Size}$ ).

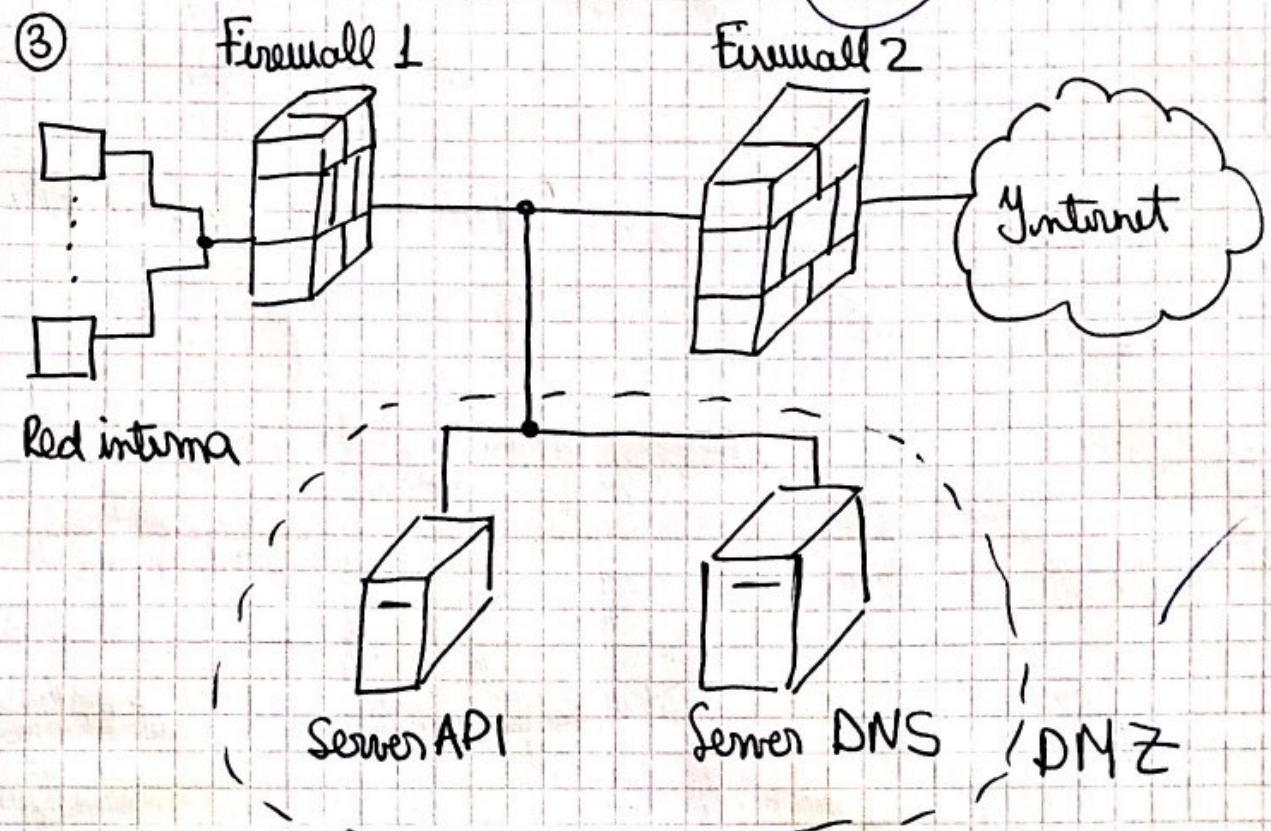
Después de  $2 \cdot \text{RTT}$ , ocurre un timeout: no se reciben los ACK en  $2 \cdot \text{RTT}$ . Por lo que se comienza con  $\text{CWND} = \text{LWS}$  y se modifica  $\text{SSTHRESH} = \text{Flight Size} / 2$ .

Nota que los bytes se cuentan desde cero de nuevo en la tabla mostrada.

La cantidad de bytes que deben retransmitirse es la cantidad perdida. Se perdieron los bytes enviados en los RTT 1 y 6:  $16 \text{ KB}$  y  $10 \text{ KB}$ . ~~Estos No.~~

Por lo tanto, deben retransmitirse  $26 \text{ KB}$ . X





### Firewall 1

Statefull, y con DROP como política por defecto.

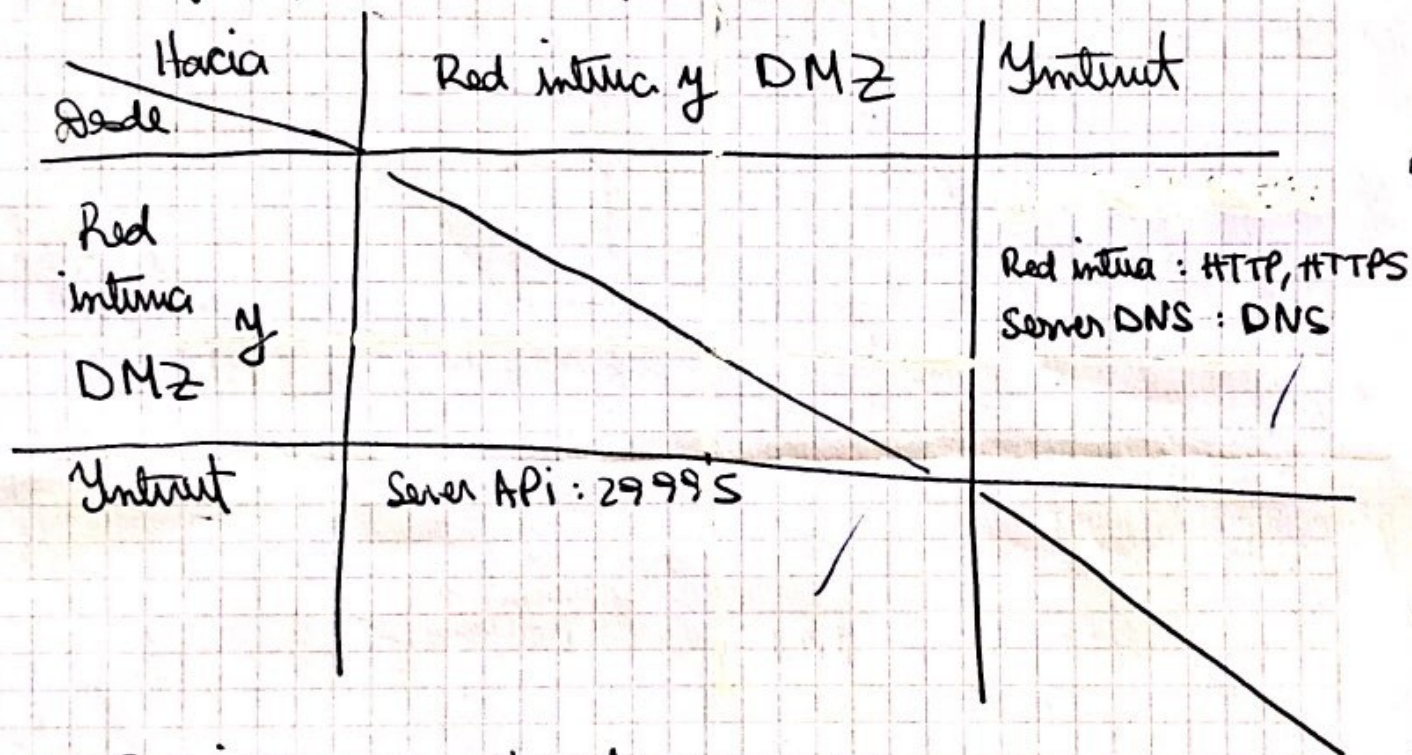
Hacia / Desde	Red interna	"Afuera"
Red interna		Internet: HTTP, HTTPS API: 29995 DNS Resolver: DNS
"Afuera" (DMZ + Internet)	DROP	



- < Red interna, \*, Ynternet, 80, TCP > (HTTP)
  - < Red interna, \*, Ynternet, port(HTTP), TCP > (HTTPS)
  - < Red interna, \*, Server API, 29995, TCP > <sup>supongamos TCP para API</sup>
  - < Red interna, \*, DNS Reshen, 53, UDP > (DNS)
- ↑ (creo que me 53 para DNS)

## Firewall 2

Statifull, con DROP por defecto.



- < Red interna, \*, Ynternet, 80, TCP >
- < Red interna, \*, Ynternet, port(HTTPS), TCP >
- < Server DNS, \*, Ynternet, 53, UDP >
- < Ynternet, \*, Server API, 29995, TCP >

(Nota: supongamos que no usamos mail en la red interna.)



b) Puede usarse un sistema tipo HMAC, en donde se manda, ~~se~~ junto con el mensaje, un hash de una clave simétrica y del mensaje. De esta forma, cuando el receptor recibe el mensaje, le suma la clave que también tiene, le aplica el hash y lo compara con el hash enviado. Como solo el servidor DNS y el receptor conocen la clave, debe ser él. ~~esto~~ ~~esto~~ Esto requiere que las partes compartan una clave previamente.

Si no es el caso, puede usarse firma digital con claves pública y privada del DNS. Esto también garantiza no repudio:  $K^- (\text{Hash}(m)) + m$ .

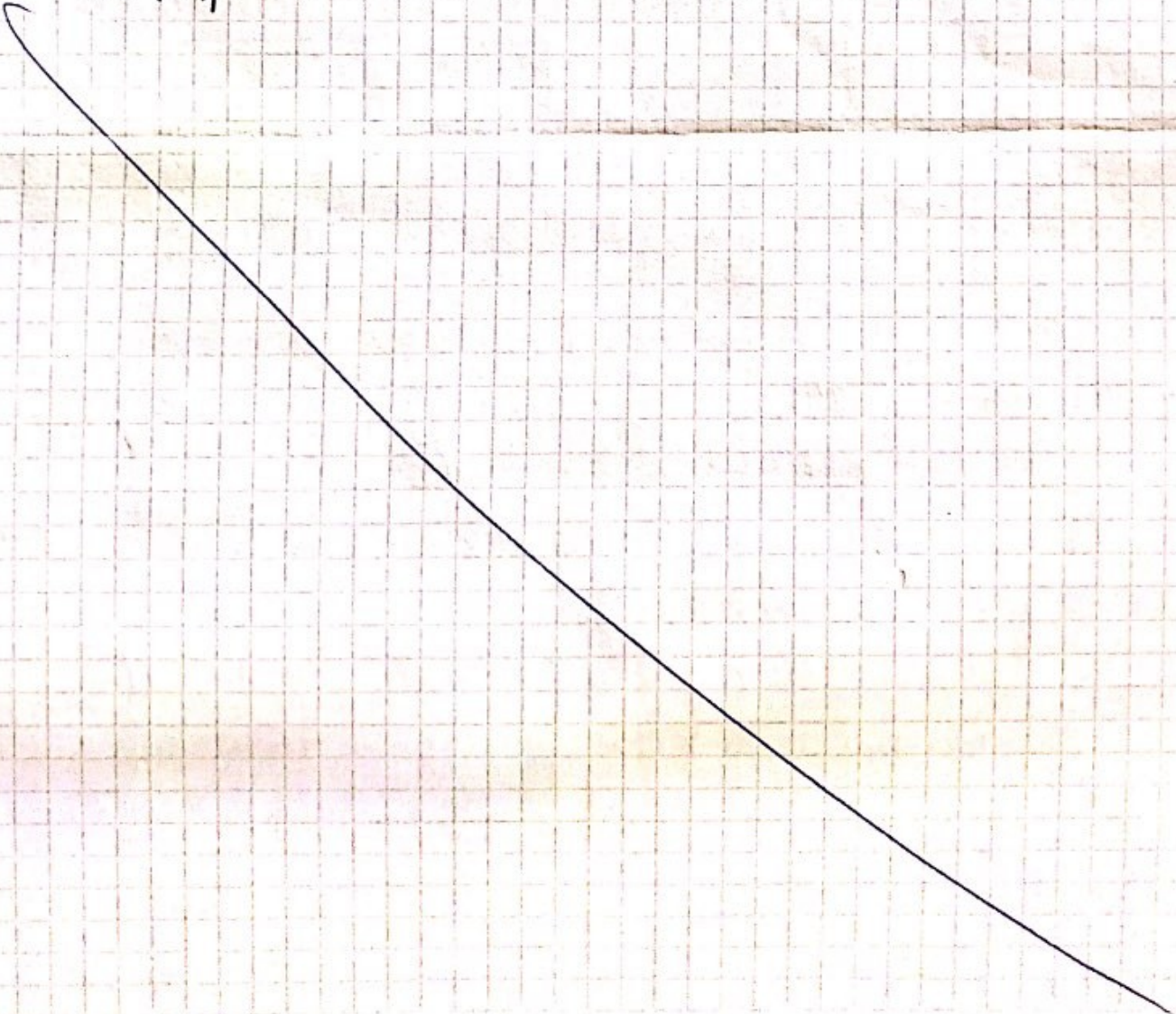
Nótese que esto también garantiza integridad.

Este hash descrito debería ir junto con el mensaje que envíe el DNS.

~~Podría tenerse un DNS seguro~~

Podría montarse un protocolo que implemente firma digital y usarlo para mandar DNS. En tal caso, debería certificarse la clave pública del servidor con alguna institución para que quien firma usan el servidor pueda obtenerla o sea segura/certificada.

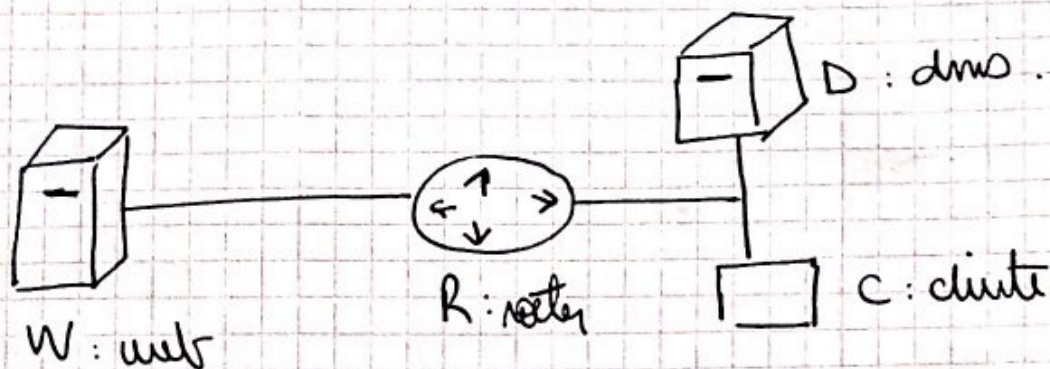


- ← (Sumando firma digital:  $K^-(\text{Hash}(m)) + m$ )
- c) • Confidencialidad: NO. El mensaje se manda a lado del hash ~~en~~ encriptado, sin encriptar.
- Integridad: SI. Se compara el hash con el mensaje.
  - No repudio: SI. Solo el remitente conoce  $K^-$ , por lo que es el único capaz de generar el paquete.
  - Autenticación: SI (lo pedía el ítem).
  - Disponibilidad: NO. Se pueden estar "dopeando" los paquetes maliciosamente.
- 



④  $http://www.ar/index.html$  en W.

B+



Asumos que:

- Los enlaces son segmentos Ethernet.
- Todos tienen su caché ARP vacía.
- D tiene  $www.ar$  en la caché.
- El cliente tiene las cachés vacías.
- No se genera tráfico en un segmento TCP.

a) mensajes de aplicación:

Origen	Destino	Protocolo	Contenido de mensaje
C W C	D W C	DNS DNS HTTP HTTP	Request: $www.ar.A$ Response: "ip" $www.ar.$ Request: GET $index.html$ Response: 200 OK "DATA"



b) Mensajes de transporte.

IP: punto Origen	IP: punto Destino	Protocolo	Mensaje
{ C:*	D:53	UDP	"Datos req. DNS"
{ D:53	C:*	UDP	"Datos resp. DNS"
C:*	W:80	TCP	[SYN]
W:5	C:*	TCP	[SYN+ACK]
C:*	W:80	TCP	[ACK] "Datos de req. HTTP"
W:80	C:*	TCP	[ACK] "Datos resp."
W:80	C:*)	TCP	[FIN]
C:*	W:80	TCP	[FIN+ACK]
W:80	C:*	TCP	[ACK]

c)

MAC Origen	MAC Destino	IP Origen	IP Destino	Prot.	Mensaje
{ C	Broadcast	—	—	ARP	who is "it" ✓
{ D	C	—	—	ARP	is at ✓
{ C	D	C	D	IP	"Datos" ✓
{ D	C	D	C	IP	"Datos" ✓
C	Broadcast	—	—	ARP	who is "it" ✓
R	C	—	—	ARP	is at ✓
C	R	C	W	IP	"Datos" ✓
R	R	—	—	ARP	who is "it W" ✓
C	Broadcast	—	—	ARP	is at ✓
W	R	C	W	IP	"datos" ✓
R	R	W	C	IP	"datos" ✓
W	R	W	C	IP	"datos" ✓
R	C	W	C	IP	"datos" ✓
C	R	W	W	IP	"datos" ✓
R	C	W	C	IP	"datos" ✓



Los paquetes marcados con  $\oplus$  se repiten 1,5 veces por el protocolo TCP que tiene a su vez un "ida y vuelta" de mensajes de ~~re~~ transporte del cliente al servidor W.

Nota que se no involucra ARP porque los ~~casos~~ ~~casos~~ ~~complejan~~ ~~casos~~, y los paquetes ARP llevan información de red. (IPs.)

Es importante mencionar que cuando C hace "who is", ya todos los demás (D y R) conocen la IP de C por lo que no tienen que hacer "who is" ellos luego. Lo mismo pasa con el "who is" que envía R preguntando por W.

